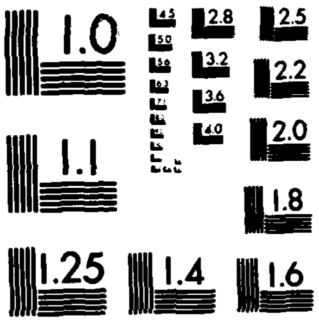
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

3

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0292	2. GOVT ACCESSION NO. AD-A127487	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMPETITION AGAINST FALLIBLE OPPONENTS		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Andrew L. Reibman and Bruce W. Ballard		8. CONTRACT OR GRANT NUMBER(s) AFOSR-81-0221
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department, Duke University, Durham NC 27706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences, Air Force Office of Scientific Research, Bolling AFB DC 20332		12. REPORT DATE Dec 1982
		13. NUMBER OF PAGES 14
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES
Submitted to the 21st Southeast Region Conference of the ACM.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Minimax-backup; game trees; search strategies; predicting opponent play.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
Most previous research on search for game playing has focused on improving search efficiency rather than on better utilizing available information. By developing models based on a notion the authors call "playing strength", they acquire the insight needed to develop strategies which perform better than minimax against both perfect and imperfect opponents. In particular situations, the authors' decision strategies yield improvements comparable to or exceeding those provided by an additional ply of search.

DD FORM 1 JAN 73 1473

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

83 04 28 063

AFOSR-TR- 83-0292

Competition Against Fallible Opponents

Andrew L. Reibman

Bruce W. Ballard

Dept. of Computer Science
Duke University
Durham, NC 27706

Paper Prepared for:

Twenty-first Southeast Region ACM Conference

April 7-9 1983
Durham, NC

December 1982

A

This Research has been sponsored by the Air Force Command, Air Force Systems Command under Grant AFOSR 81-0221.

83 04 28 063

Abstract

Most previous research on search for game playing has focused on improving search efficiency rather than on better utilizing available information. By developing models based on a notion we call "playing strength", we acquire the insight needed to develop strategies which perform better than minimax against both perfect and imperfect opponents. In particular situations, our decision strategies yield improvements comparable to or exceeding those provided by an additional ply of search.

1. Mini-max for game playing

In many competitive situations, decision making can be aided by the use of game models Any two-player, zero-sum game can be represented as a minimax game tree, where the root of the tree denotes the initial game situation and the children of any node represent the results of the possible moves which can be made from that node. In this paper we consider ways of improving the performance of the standard minimax backup algorithm. We follow convention and call the two players "Max" and "Min" and use "+" to denote nodes where Max moves and "-" to represent similar nodes for Min. Positive endgame (leaf) values denote positive payoffs for Max. Readers unfamiliar with the conventional minimax backup search and decision procedure should refer to Nilsson [80].

2. Problems with minimax

Given perfect play by our opponent, we know from game theory that a conventional minimax strategy which searches the entire game tree yields the highest possible payoff. However, most actual players, whether human or machine, lack the conditions needed to insure optimal play. In particular, because the trees of many games are very deep, and tree size grows exponentially with depth, a complete search of most real game trees is computationally intractable. In these instances, static evaluation functions and other heuristic techniques are employed to reduce the search used in making decisions.

Most previous research on search for game playing has focused on improving search efficiency. Results of this type improve the quality of player decision making by providing more relevant information. In contrast, our research focuses on better utilizing information rather than searching for more. We summarize previous work on this issue, then describe our approach and provide initial results.

2.1 Previous work on compensating for incomplete search

During the middle to late 1960's, James Slagle and his associates sought to improve the performance of minimax backup by attempting to predict the expected value of (D+1)-level minimax search with only a D-level search (Slagle and Dixon [70]). Their strategy was called the "M and N procedure" and determined the value of a Max node by considering its M best children and the value of a Min node from its N best children. The study cited above restricted the problem to search depth D=2 and also restricted M and N to be 2. The algorithm they devised, which we summarize below, was tested on actual trees arising in the game of Kalah.

The M and N procedure is based on the notion that the expected backed-up value of a node is likely to differ from the expected backed-up value of its best child. To investigate the exact nature of this difference, investigators generated sample Kalah positions and ran 1-ply and 2-ply searches on them. From this empirical data they plotted the difference between the static and 2-ply backed-up values of each position against the difference between the static values of the two best children. From this data they defined a "bonus function" to be added to the static value of best-looking child, hoping that this would lead to a better estimate of the true value of the parent. As shown in their paper, this bonus function turned out to be approximately linear. Their results were (1) that the "improved" algorithm won about 51.1 percent of the games, and (2) that M and N yields an improvement in the expected value of the outcome of the game about 13 percent as great as does an additional ply of search. They conclude that "M and N as applied to Kalah provides an advantage that is about as large as a typical value feature [of the static evaluation function] but not as large as an unusually powerful one".

The reader will note in the following section a resemblance between the notion of a bonus function and our attempt to more accurately predict the expected value of moves made by a fallible opponent. In Ballard and Reibman [83] we prove that in the simplest form of the first of two models we present below, with a fixed probability of opponent error, independent of the difference between candidate nodes, our results can be obtained by an appropriate form of M and N strategy (and vice versa), although the exact backed-up values being determined will differ. This is due to the linear nature of the bonus function determined as outlined above. We have also shown that the arc-sum tree model we have adopted would lead not to a linear bonus function but rather to one described by a 4-th degree polynomial.

2.2 The unresolved problem of opponent fallibility

In addition to having an inability to completely search actual game trees, actual implementations of minimax assume perfect play by their opponent. However, this assumption often is overly conservative and can be detrimental to good play. An immediate and extreme example is found in forced loss situations. In the two-valued game in Figure 1, Max is faced with a forced loss. Regardless of the move Max makes at the "+" node, if Min plays correctly Max will always lose. Following the conventional minimax strategy, Max would play randomly, picking either subtree with equal frequency. Suppose, however, that there is a nonzero probability that Min will play incorrectly. For the moment, assume Min makes an incorrect move 10% of the time. Then if Max moves randomly, the expected outcome of the game is .5(0) +

.5(.9*0 + .1*1) = .05. If Max knows that, on occasion, Min will move incorrectly, this knowledge can be used to improve the expected payoff from the game. Specifically, Max can regard each "-" node as a "chance node" similar to those that represent chance events such as dice rolls in non-minimax games. (Ballard [82,83] gives algorithms suited to this broader class of "*-minimax" games.) Thus Max evaluates "-" by computing a weighted average of its children based on their conjectured probabilities of being chosen by Min rather than by finding just the minimum. Following this strategy, Max converts the pure minimax tree of Figure 1 to the *-minimax tree shown in Figure 2, determines the values of the children of the root as 0 and 0.1, and selects the rightmost branch of the game tree because it now has the higher backed-up value. In terms of expected payoff, (which is computed as 0*(0) + 1.0*(.9*0 + .1*1) = 0.1), this is clearly an improvement over standard minimax play. Furthermore, this strategy is an improvement over minimax in forced loss situations regardless of the particular probability that Min will err. Our observed improvement in forced loss situations is a specific example of "tie-breaking", where the equal grandchild values happen to be zero. Because minimax only uses information provided by the extreme-valued children of a node, positions with different expected results often appear equivalent to minimax. Variant minimax strategies can thus improve performance by breaking ties with information minimax obtains but does not use.

A less obvious opportunity for improving minimax's performance is found in Figure 3. Assume as above that Min makes the correct move with probability .9. If Max uses the conventional backup strategy and chooses the left node, the expected outcome of the game is 2.1. If, however, we recognize our opponent's fallibility and convert the Min nodes to "*'s", (as in Figure 4), we will choose the right branch and the games expected result increases to 2.9. Thus by altering the way we backup values to our opponent's nodes in the game tree, we can improve our expected performance against an imperfect opponent.

In the example of a forced loss, the improvement in performance was due to the ability of a weighted average backup scheme to correctly choose between moves which appear equal to conventional minimax. In the second example, our variant backup yielded a "radical difference" from minimax, a choice of move which differed not because of "tie-breaking", but because differing backup strategies produced distinct choices of which available move is correct.

3. Adversary models for investigation

Having observed an opportunity to profit by exploiting errors which might be made by our opponent, we now formulate a model of a fallible adversary's behavior. Our model is based on the concept we call "playing strength". Intuitively, playing strength is an indication of how well a player can be expected to perform in actual competition rather than against a theoretical perfect player. In order to be a useful metric, a playing strength parameter should have at least the following properties:

I. Given two players M and N, where the playing strength of M greatly exceeds that of N, M should play better than N against any fixed third player Q, where better play is defined as winning a higher proportion of the games played or having a better expected payoff.

II. Assume M and N are as above and consider a node P where M and N have two possible moves available. If the children of P are denoted r and s, and the expected payoff from making move r is greater than the expected payoff from making move s for both M and N, then M should choose move r over move s with a probability greater than or equal to the probability that N chooses r over s.

3.1 A simple playing strength based model

Having given general axioms for our intuitive notion of playing strength, we now describe a particular model for an imperfect player's behavior we have chosen to study. Actually, we have already presented a simple example of playing strength in a model for playing binary-tree games. Let the playing strength be the probability that a player chooses the move which, in the present game position against the current opponent, yields the best expected endgame result. Since the theoretically correct move can be difficult to determine, we approximate it in our model by using a conventional minimax search. Using this approximation, we model an imperfect Min player of strength S by choosing the move with the best minimax evaluation with probability S and the other available move with probability 1-S. We can generalize this model for trees with branching factor B in several ways: (1) by considering only the two best available moves; (2) by choosing the first move with probability S, the second with probability S*(1-S), the third with probability S * (1-S)**2, and the nth with probability S * (1-S)**(B-1); (3) by using other variant decision strategies not discussed here. In the second case, it should be noted that these probabilities are an approximation. Their sum approaches 1 only as B goes to infinity, otherwise the sum differs from 1 by (1-S)**B.

3.2 A more sophisticated model of imperfect play

The model presented is simple to implement and, as shown in Ballard and Reibman [83], results in better play than does minimax in some broad classes of game situations. However, it fails to consider the relative differences between moves. For example, if node values range from 0 to 10, any reasonable player should choose a node valued 2 over a node valued 10 more often than a node valued 2 would be chosen over one valued 3. To correct this we present a second model of imperfect play.

In general, it should be fairly easy to differentiate between moves whose values differ greatly. However, if two moves have approximately the same value, it could be a more difficult task to choose between them. The strength of a player is, in part, his ability to choose the correct move from a range of alternatives. Playing strength can therefore correspond to a "range of discernment", the ability of a player to determine the relative quality of moves. An inability to distinguish between moves with radically different expected outcomes could have drastic consequences. Similar difficulties with moves of almost equal expected payoff should, on the average, have much less effect on a player's overall performance.

We model players of various strengths by adding noise to the information they use for decision making. A player with noiseless move evaluation is a "perfect opponent", while a player with an infinite amount of noise injected into its evaluation plays randomly. We introduce noise at the top of an imperfect player's search tree in an amount inversely proportional to the player's strength. Although it may appear to be easier, and perhaps more reasonable, to introduce noise at the leaves or in the static evaluation function, we avoid this alternative for a number of reasons. First, we feel introducing noise at the top better models the notion of opponent fallibility while noise in the leaves reflects the problems of incomplete search. Furthermore, the actual effect of adding noise to the tops of search trees can be studied analytically while the effects of introducing noise in the leaves are not yet understood [Nau 80]. We now describe the details of the imperfect player model used in the remainder of this paper. Each imperfect player is assigned a playing strength between 0 (perfect) and minus infinity (random). In our simulation, the imperfect Min player conducts a conventional minimax backup search to approximate the actual value of each child of the current position. The backed-up values of each child are then normalized with respect to the range of possible backed-up values and a random number, 0 <= x <= -S, (where S is the playing strength, a real number <= 0), is added to the normalized value of each child. The true value with noise added is then treated as a conventional backed-up value.

4. An empirical analysis of the effect of imperfect play

In order to investigate the correlation between playing strength as defined in our model and performance in actual competition, we have conducted trials pitting minimax against imperfect opponents with varying playing strengths. We conduct our trials with complete, n-ary game trees generated as functions of three parameters: D denotes the depth of the tree in ply, Br the branching factor, and V, the maximum allowable "arc value". In previous studies of search for game playing, leaves have been assigned independent random numbers as values (e. g. Knuth and Moore [],Pearl [83]), or their values have been obtained by growing the tree in a top-down fashion (Fuller, et al []). In our empirical study, we employ the latter method. Every arc in the tree is assigned a random integer chosen from a uniform distribution between 0 and V. The value of each leaf is then the sum of the arcs leading to it from the root. This method insures a fairly strong dependence between the values of brother and sisters in the game tree. (We are in the process of formulating a method of characterizing the actual *degree of dependence* in game trees.) We feel that a game with some dependence more accurately models real world applications of game playing and reduces the chance of anomalous behavior (Pearl [82]). In addition to producing trees with a fairly high degree of dependence, the method of top-down tree growth has two other advantages. First, if the tree is fairly deep in relation to the players' search depths, we need to grow only those paths which immediately surround the line of play, a great savings in simulation time. Second, the arc-sum *method of* leaf value calculation provides a natural static evaluation function for a node, the sum of the arcs leading from the node to the root.

We have conducted a number of experiments to measure the gains made by minimax against imperfect opponents of varying strengths. We present the results of three such experiments, each consisting of 1000 game trees with D=5, V=10, and Br=2, 4, or 10. Both Max and Min used 2-ply searches and the partial arc-sum static evaluation to determine their moves. Thus, each game lasted five moves, and the Min player had the first opportunity to see the actual leaves of the game tree. The trees were created by generating random arc values with the UNIX pseudo-random number generator on a PDP-11/70. For each collection of 1000 trees, Max played each game against several opponents with differing playing strengths. The results are summarized in Figure 5. As expected, Max's payoff increased monotonically as the imperfect player model's strength decreased.

5. A strategy for use against an imperfect opponent

We now present a strategy based on the *-minimax search algorithms for trees containing chance nodes in order to improve performance by compensating for the probabilistic behavior of a fallible opponent. Our strategy predicts Min play by using the following assumptions to evaluate "-" nodes:

I. Against a Min player assumed to be perfect, we should use a conventional Max strategy.

II. Against an opponent who is assumed to play randomly, we should evaluate "-" nodes by taking an unweighted average of the values of their children. values of their children.

III. In general, against imperfect players, we should evaluate "-" nodes by taking a weighted average of the values of their children, deriving the appropriate probabilities for computing this average by using, in part, an estimate of our opponents playing strength.

To predict the moves of our imperfect opponent, we consider the *-minimax based model of imperfect player behavior presented in section 3-1. More specifically, we assign our opponent a predicted strength, denoted PS, between 0 and 1. To determine the value of "-" nodes directly below the root, our predictive strategy searches and backs up values to the nodes directly below each "-" node using conventional minimax. Each "-" node is then evaluated by first sorting the values of its children in increasing order, then taking a weighted average using probabilities PS, (1-PS)*PS,...,(1-PS)**(Br-1) * PS. If PS=1, we have predicted that our opponent is perfect, so we consider only the minimum-valued child in evaluating a "-" node. At the other extreme, if a random opponent is predicted, i. e. PS is approximately 0, the probabilities used to compute the weighted average are all equal and the Min node is evaluated by averaging the values of its children.

How well our assumptions predict the moves of imperfect opponents should be reflected in our strategy's actual performance against such players. Note that the playing strength metric as used in the simulated Min player may not directly correspond to the playing strength metric used by our Max player to predict Min's behavior. We are currently investigating how to choose the predicted strength which yields the maximum payoff for our strategy given an opponent model and an actual playing strength.

6. An empirical analysis of predictive play

Against imperfect players of selected strength we conduct an empirical study to compare the performance of our predictive algorithm with that of conventional minimax backup. As in the empirical analysis in section 4, we use a sample of 1000 randomly generated game trees with Br=4, D=5, and V=10. We use three Min opponents: true Min with no noise added, an imperfect Min player with noise values obtained from a uniform distribution between 0 and .5, and an approximation of a random player with noise values chosen from the range 0 to 6. Against these Min players, we test 1-, 2-, and 3-ply searching conventional Max and 10 predictive players, each with a 2-ply search and and a PS chosen from between 0 and .9. The results of this experiment are found in Figure 6. Before summarizing our observations, we note that the numbers given in Figure 6 represent points on a continuum; they indicate general trends but do not convey the entire spectrum of values which lie between the points we have included.

In the first column of Figure 6, we observe that, though it might be expected that pure Max backup would be the optimum strategy against conventional Min, several of our predictive players perform better than a conventional Max player searching the same number of ply. Our observed improvement is as much as 7% of the gain we would expect from adding an additional ply of search to the conventional Max strategy. This result is analogous to that obtained with Slagle and Dixon's M and N strategy. Like M and N, our improvement is due, at least in part, to a strategy which, by considering information from more than just the extreme-valued children of a node, partially compensates for a search which fails to reach the leaves.

In the second column of Figure 6, we see that against an opponent whose play is sometimes imperfect, our strategy can provide almost half the expected improvement given by adding an additional ply of search to a conventional Max strategy. We believe this gain is due primarily to the ability of our strategy to capitalize on our opponents potential for errors.

If we examine the results in column 3 of Figure 6, we observe that, against a random player, our strategy yields an improvement between 2 and 3 times that *provided* by an additional ply of search. As the predicted strength of our opponent goes down, our predictions of our opponent's moves become more a simple average of the alternatives available to him than a minimax backup. We have previously conjectured that the most accurate prediction of the results of random play is such an average and, as expected, our strategy's performance continues to improve as the strength predicted decreases.

Our final comment is to observe a possible drawback to the indiscriminate use of our strategy. When we begin to overestimate our opponents fallibility, our performance degrades. In both columns 1 and 2, our performance peaks, then if we inaccurately overestimate the weakness of our opponent, our performance declines and, in column one, actually falls below that of conventional Max.

7. Conclusion

In this paper we have introduced the problem of adapting game playing strategies to deal with imperfect opponents. We first observed that, against a fallible adversary, the conventional minimax backup strategy does not always choose the move which yields the best expected payoff. To investigate ways of improving minimax, we formulated a general model of an imperfect adversary using the concept of "playing strength". We then proposed several alternative game playing strategies which capitalize on their opponents potential for error. An empirical study was conducted to compare the performance of these strategies with that of minimax. Even against perfect opponents, our strategy showed a marginal improvement over minimax and, in some other cases, great increases in performance were observed.

We have presented the preliminary results of our efforts to develop variant minimax strategies that improve the performance of game players in actual competition. Our present and future research includes a continued effort to expand and generalize our models of imperfect play, our predictive strategy, and the notion of playing strength. Further study of our models has included not only additional empirical experiments but also closed-form analysis of some closely related game tree search problems. We hope to eventually acquire a unified understanding of several distinct problems with minimax in order to develop a more general game playing procedure which retains the strong points of minimax while correcting its perceived inadequacies.

References

Ballard, B. A Search Procedure for Perfect Information Games of Chance: Its Formulation and Analysis. Proceedings of AAAI-82, August 1982.

Ballard, B. The *-Minimax Search Procedure for Trees Containing Chance Nodes. Artificial Intelligence, to appear.

Ballard,B. and Reibman, A. What's wrong with minimax? Paper in preparation.

Fuller, S., Gaschnig, J. and Gillogly, J. An analysis of th alpha-beta pruning algorithm. Dept. of Computer Science Report, Carnegie-Mellon University, July 1973.

Knuth,D and Moore, J. An analysis of alpha-beta pruning. Artificial Intelligence, 6 1975, 293-326.

Nau, D. Quality of Decision Versus Depth of Search in Game Trees: A summary of Results. First Annual National Conference on Artificial Intelligence, August 1980.

Nau, D. Pathology on Game Trees Revisited and an Alternative to Minimaxing. Technical Report, Dept. of Computer Science, University of Maryland, Jan. 1982.

Nilsson, N. Principles of Artificial Intelligence. Tioga, Palo Alto, 1980.

Pearl, J. On the Nature of Pathology in Game Search. Technical Report UCLA-ENG-CSL-82-17, School of Engineering and Applied Science, University of California, Los Angeles, Jan. 1982.

Pearl, J. The Solution of the Branching Factor of the Alpha-Beta Pruning Algorithm and its Optimality. Communications of the ACM, August 1982, 25 (8), 559-563.

Slagle, R. and Dixon, J. Experiments with the M & N Tree-Searching Program. Communications of the ACM, March 1970, 13 (3), 147-153 .

Figures

```
                    +
 backup           /   \       backup
 value=0         /     \     value=0
                /       \
               -         -
 p=.5  / \ p=.5   p=0  / \ p=1
      /   \           /   \
     0     0         1     0
```

Figure 1: Max faced with a forced loss

```
                    +
 backup           /   \       backup
 value=0         /     \     value=.1
                /       \
               *         *
 p=.5  / \ p=.5   p=.1 / \ p=.9
      /   \           /   \
     0     0         1     0
```

Figure 2: This *-minimax tree is a variant of the pure mini-max tree in Figure 1 given the assumption that min will err 10% of the time.

```
                    +
 backup           /   \       backup
 value=2         /     \     value=1
                /       \
               -         -
 p=1   / \ p=0    p=1  / \ p=0
      /   \           /   \
     2     3         1     20
```

Figure 3: Pure Max chooses the left branch.

```
                      +
  backup       /   \       backup
  value=2.1 /         \   value=2.9
           /            \
          *               *
  p=.9 / \ p=.1   p=.9 / \ p=.1
      /   \           /   \
     2     3         1     20
```

Figure 4: This *-minimax tree is a variant of the pure mini-max tree in Figure 3. Given the assumption that min will err 10% of the time, Max will choose the right branch.

Spread of Noise Added to Min's Move Evaluation	Branching Factor 2	4	10
0.0	26.5	28.3	29.2
0.1	27.5	28.7	30.9
0.2	28.5	29.9	32.3
0.3	29.0	31.0	33.5
0.4	29.4	31.6	34.2
0.5	30.2	32.9	35.2
6.0	31.3	34.6	37.6

Figure 5: Mini-max vs. the Imperfect Player Model. This table gives the average payoff for minimax in depth 5 games against imperfect opponents of various strengths. Each player conducted a 2-ply search. The arc-values ranged from 0-10. The average is based on 1000 trial game trees.

	Spread of Noise Added to Min's Move Evaluation		
Max player	0.0	0.5	6.0
conventional 1-ply	27.31	30.37	34.26
conventional 2-ply	28.26	32.88	34.64
conventional 3-ply	29.13	33.43	34.79
predictive 2-ply			
PS=.9	28.31	33.00	34.77
PS=.8	28.31	33.00	34.78
PS=.7	28.31	33.01	34.78
PS=.6	28.32	33.01	34.79
PS=.5	28.30	33.04	34.85
PS=.4	28.30	33.07	34.88
PS=.3	28.25	33.08	34.97
PS=.2	28.20	33.10	34.98
PS=.1	28.12	33.07	35.00

Figure 6: This table compares the relative performance of conventional and predictive Max players against three Min players of different strengths. Each figure is based on 1000 game trees with D=5, Br=4, and V=10.